Assignment 1: Toy Train Set

Due: Week three. (Check the course web site for campus-specific details).

Web stuff: See the course web site for starter code, hints, and announcements about the assignment.

Purpose: Introduce simple graphics, review Java basics including methods of storing multiple objects, and review program design.

Overview: You will be given some starter code that models a simple toy train set and you will extend it to model a more involved train set. The graphics are very basic, using only ASCII characters, so you'll need some imagination when viewing the output.

Current state: The program shows trains made up of boxcars that travel back and forth on tracks. The starter code can only handle two train tracks. A track can run north/south or east/west. On each track is one train that is exactly one car long. There is only a single type of railcar, a boxcar. (Somehow it manages to chug back and forth without a locomotive.) When a train reaches the edge of the display it reverses its direction.

We designed the program to read a description of the train set from an input file. (The format of the input is described in the TrainSetDriver class.) The input file provided with the starter code describes a north/south track and train, and an east/west track and train, because that's all the current program can handle. Once you have extended the program to handle fancier train sets, you won't have to change the part of the program that reads a train set description and builds a train set — just the input file.

You may assume that there are no errors in the input.

Your task: You will extend the program so that it can handle:

- trains that contain more than one car,
- two new types of railcar in addition to boxcars: locomotives and cabooses, and
- train sets that have more than two train tracks (but still never more than one train per track).

To accomplish these tasks you may find it appropriate to add classes to the program, or to add or modify variables or methods in the existing classes.

Your task in detail: Do the assignment in the order below. Test your code as you go. For each step, you'll need to create new input files to describe the increasingly fancy trains you can handle.

1. Read the remainder of this handout. Download the starter program and add the classes to a new project. (Refer to the CodeWarrior Survival Guide on the course web site for instructions.) Download the input file into the same folder as this project. Run the program to see what it does (TrainSetDriver is the main class). Study the code provided to find out:

   - the purpose of each class,
   - how the classes relate (Does any class extend another? Does any class contain an instance of another?), and
   - which classes you will have to modify, as noted in the comments.

   Create new input files and test the program with them. Take care when choosing train start locations so the train appears on the screen.

2. Modify the program to handle trains that have any number of cars. For now, assume that all cars are boxcars. Since the number of cars in each train is known at the time you make the train (it is read from the input) the appropriate Java construct to use is an array. Although your program should be able to build trains with any number of cars, limit the input to at most six cars in order to avoid problems with trains too long to fit on the screen.

3. Modify the program to handle locomotives and cabooses, in addition to boxcars. Check the course web site for diagrams of these; you'll have to turn them into little character graphics as we did with boxcars. The dimensions of your locomotive and caboose graphics must be the same as the boxcar provided. Assume that the input file describes a a train that has one locomotive, followed by at most four boxcars, followed by a caboose.
   Hint: Think carefully about your design before writing any code.

4. Modify the program to handle a train set that has more than two tracks. The number of tracks (and hence trains) is not explicitly stated in the input, so it is not known until after you start creating tracks. A vector is therefore appropriate for storing the set of tracks.

What to submit: You will not hand in a paper submission for this assignment. All your .java files must be submitted electronically. Details including campus-specific requirements regarding due time, file names and submit location will be provided on the different campus web sites.

Graphics: See the announcements page for a brief introduction to graphics in Java. This is not required reading.

Javadoc: The starter code contains odd looking comments with words like "`@param`." Javadoc is a program that uses these to generate documentation that can be viewed on the web. See the course web site for an introduction to javadoc.
Like ours, the comments you write should use javadoc.